SPECIAL COLLECTOR'S ISSUE

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

www.linuxjournal.com

LINUX JOURNAL LITE

FREE SAMPLE

200TH ISSUE CELEBRATION!

Parallel Programming with NVIDIA | Control a Fridge with Pogoplug | How-To: PiTiVi Video Editor | 200 Things to Do with Linux | Web Retrospective and Predictions

REVIEWED ZOTAC ZBOX HD-ID11 and Barnes & Noble's Nook

DECEMBER 2010 | ISSUE 200

$5.99US $5.99CAN

12>

0 09281 03102 4

PHONE: 713.344.1956
P.O. BOX 980985 HOUSTON, TX 77098 USA
WWW.LINUXJOURNAL.COM
A BELLTOWN MEDIA PUBLICATION

This sample issue contains a few articles from the December 2010 issue of *Linux Journal*, a test drive if you will, of the full version of the magazine. (See pages 3 and 4 of the PDF for the full issue table of contents.)

We hope you'll take this opportunity to enjoy the award-winning content we've become known for, and if you like what you see, order your own subscription to *Linux Journal*. Subscribers save up to 75% off the newsstand price, receive a free gift, and get the magazine conveniently delivered to them well before it hits newsstands.

In this edition of *Linux Journal Lite*, we feature the following selections:

If you use Linux, you should be reading *Linux Journal*. Subscribe today: www.LinuxJournal.com/subscribe.

We hope you enjoy this sample issue!

Sincerely,

Carlie Fairchild, Publisher

CONTENTS DECEMBER 2010 Issue 200

200th ISSUE CELEBRATION

FEATURE

ON THE COVER

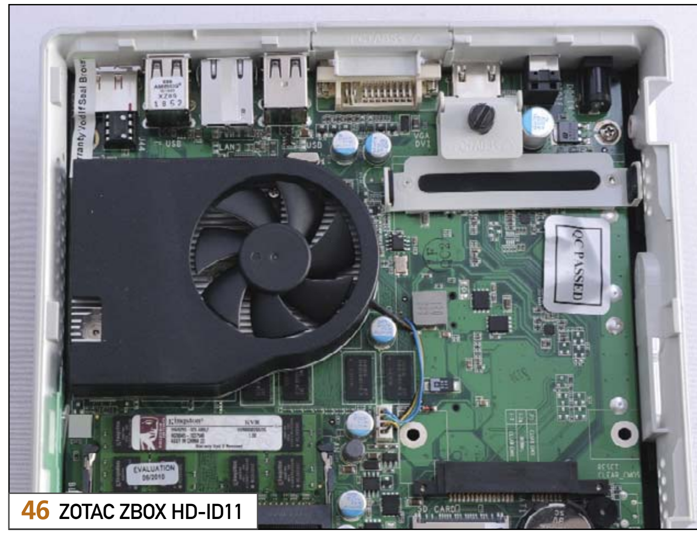
CONTENTS DECEMBER 2010 Issue 200

COLUMNS

REVIEWS

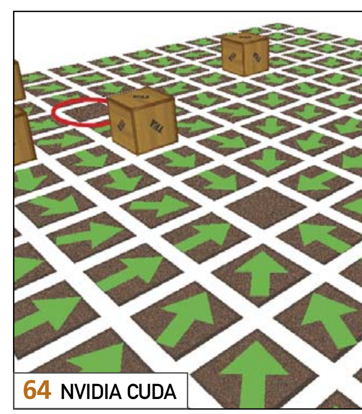
INDEPTH

IN EVERY ISSUE

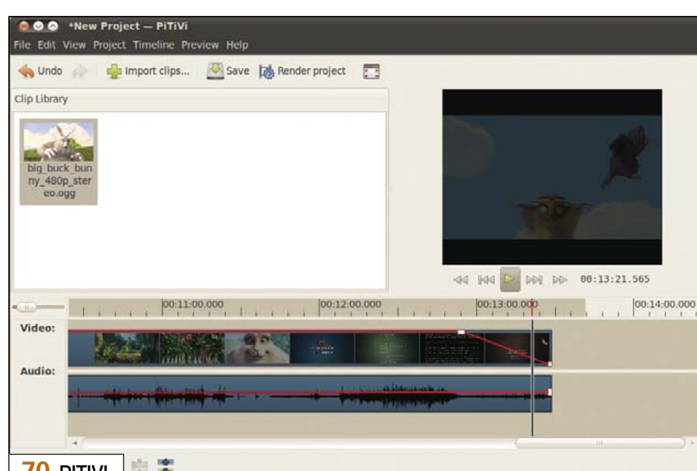


46 ZOTAC ZBOX HD-ID11

48 BARNES & NOBLE'S NOOK



64 NVIDIA CUDA



70 PITIVI

USPS *LINUX JOURNAL* (ISSN 1075-3583) (USPS 12854) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 310, Houston, TX 77056 USA. Periodicals postage paid at Houston, Texas and at additional mailing offices. Cover price is $5.99 US. Subscription rate is $29.50/year in the United States, $39.50 in Canada and Mexico, $69.50 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 16476, North Hollywood, CA 91615. Subscriptions start with the next issue. Canada Post: Publications Mail Agreement #41549519. Canada Returns to be sent to Pitney Bowes, P.O. Box 25542, London, ON N6C 6B2

At Your Service

MAGAZINE

PRINT SUBSCRIPTIONS: Renewing your subscription, changing your address, paying your invoice, viewing your account details or other subscription inquiries can instantly be done on-line, **www.linuxjournal.com/subs**. Alternatively, within the U.S. and Canada, you may call us toll-free 1-888-66-LINUX (54689), or internationally +1-818-487-2089. E-mail us at subs@linuxjournal.com or reach us via postal mail, Linux Journal, PO Box 16476, North Hollywood, CA 91615-9911 USA. Please remember to include your complete name and address when contacting us.

DIGITAL SUBSCRIPTIONS: Digital subscriptions of *Linux Journal* are now available and delivered as PDFs anywhere in the world for one low cost. Visit **www.linuxjournal.com/digital** for more information or use the contact information above for any digital magazine customer service inquiries.

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at **www.linuxjournal.com/contact** or mail them to Linux Journal, PO Box 980985, Houston, TX 77098 USA. Letters may be edited for space and clarity.

WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line, **www.linuxjournal.com/author/index**.

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line, **www.linuxjournal.com/advertising**. Contact us directly for further information, ads@linuxjournal.com or +1 713-344-1956 ext. 2.

ON-LINE

WEB SITE: Read exclusive on-line-only content on *Linux Journal*'s Web site, **www.linuxjournal.com**. Also, select articles from the print magazine are available on-line. Magazine subscribers, digital or print, receive full access to issue archives; please contact Customer Service for further information, subs@linuxjournal.com.

FREE e-NEWSLETTERS: Each week, *Linux Journal* editors will tell you what's hot in the world of Linux. Receive late-breaking news, technical tips and tricks, and links to in-depth stories featured on **www.linuxjournal.com**. Subscribe for free today, **www.linuxjournal.com/enewsletters**.

LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Associate Editor	Mitch Frazier mitch@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors
Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan

Proofreader Geri Gale

Publisher	Carlie Fairchild publisher@linuxjournal.com
General Manager	Rebecca Cassity rebecca@linuxjournal.com
Senior Sales Manager	Joseph Krack joseph@linuxjournal.com
Associate Publisher	Mark Irgang mark@linuxjournal.com
Webmistress	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

***Linux Journal* is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

Advertising
E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
PHONE: +1 818-487-2089
FAX: +1 818-487-4550
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 16476, North Hollywood, CA 91615-9911 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Filesystem hints are attributes that filesystems can pass down to storage devices. The devices then use the hints to make decisions about how to lay down their data most efficiently. **Matthew Wilcox** has expressed interest in implementing this. Specifically, he wanted to implement the **NVMHCI working group**'s recommended set of filesystem hints. But, as **James Bottomley** pointed out, the filesystem and the hardware had no real basis for agreement on what any given hint actually meant. So the filesystem would make guesses about what kind of hints to give the device, and the device would make guesses about what those hints actually meant. As James said, one of the most interesting things is that systems using filesystem hints seem to do better than those that don't, in spite of the guesswork involved. But, folks like **Alan Cox** remain unconvinced, saying he'd bet a beer on the fact that filesystem hints would end up not being used, even if they were fully implemented in the kernel. He didn't see enough benefit.

Robert P. J. Day has started offering **kernel programming classes**. See **www.crashcourse.ca/wiki/index.php/Online_beginner's_kernel_programming_course** for details. Some of the lessons are available for free; others are available at what seems like a pretty low fee. I haven't taken the class myself, and I'm not getting a kickback for mentioning it, but it seems like an interesting way for folks to get started with kernel hacking.

Using **kernel-level encryption** can be slow, but several folks, including **Miloslav Trmac**, recently argued that it would protect user-space applications from certain kinds of malicious attacks. Miloslav submitted a patch implementing a user-space interface to the kernel's encryption routines. This inspired a number of complaints. **Theodore Y. Ts'o** felt the speed issues would be pretty significant, and he wanted to make sure that potential users were made well aware of the huge slowdown their code would experience if they used this API instead of a user-space implementation of the same basic feature. **Arnd Bergmann** also found Miloslav's code to be overly complex, but this was explained by the fact that so many user requests had come in for extensions to Miloslav's initial implementation. The complexity was necessary to accommodate those requests. In spite of the general complaints against this code, it does seem as though the security reasons do justify it, so none of the critics seem to be objecting too loudly. I'd expect a clean implementation to make it into the kernel.

There was a bit of a scare recently when **Linus Torvalds** received a set of patches that appeared not to have been compiled or tested at all, in spite of the long "Signed-Off-By" chain listed in the patch e-mail messages. One of the main values of the "lieutenant" system is that patches are vetted through a series of trusted people who understand what Linus wants and can give it to him. If that system ever broke down, Linus probably would have to fall back on the "maintainer" system, which would be less good, because maintainers often are selected based solely on their willingness to do that job, and not on their specific reliability as producers of Linus-worthy code. The lieutenant system, in part, helps communicate various requirements to the maintainers. In this particular case, **Len Brown** had done an incorrect merge between some ACPI branches and then fed the wrong branch of his tree into his test suite. It's a very unusual confluence of errors, but the result was that some patches made it to Linus without the proper testing—just one of those things that happens and gets fixed.

—ZACK BROWN

They Said It

Any program is only as good as it is useful.
—Linus Torvalds

I like to think that I've been a good manager. That fact has been very instrumental in making Linux a successful product.
—Linus Torvalds

Making Linux GPL'd was definitely the best thing I ever did.
—Linus Torvalds

Before the commercial ventures, Linux tended to be rather hard to set up, because most of the developers were motivated mainly by their own interests.
—Linus Torvalds

Microsoft isn't evil, they just make really crappy operating systems.
—Linus Torvalds

When you say "I wrote a program that crashed Windows", people just stare at you blankly and say "Hey, I got those with the system, for free."
—Linus Torvalds

The cyberspace earnings I get from Linux come in the format of having a Network of people that know me and trust me, and that I can depend on in return.
—Linus Torvalds

People enjoy the interaction on the Internet, and the feeling of belonging to a group that does something interesting: that's how some software projects are born.
—Linus Torvalds

Non-technical questions sometimes don't have an answer at all.
—Linus Torvalds

Software is like sex: it's better when it's free.
—Linus Torvalds

The memory management on the PowerPC can be used to frighten small children.
—Linus Torvalds

MICK BAUER

Building a Transparent Firewall with Linux, Part IV

Arm your stealth firewall with a custom iptables script.

I've been writing a multipart series on building a transparent (bridging) firewall using Linux. Specifically, I'm using the distribution OpenWrt running on a Linksys WRT54GL broadband router, a hardware choice driven mainly by my curiosity about the WRT54GL's built-in five-port Ethernet switch and its ability to run OpenWrt Linux.

So far I've covered installing OpenWrt, recompiling a new OpenWrt image with iptables' bridging functionality enabled and configuring networking using OpenWrt's uci (Unified Configuration Interface) command.

This month, I review the example network topology and finally begin configuring iptables, the heart of the whole undertaking. Before I do so, however, there are a few OpenWrt housecleaning tasks to get out of the way: tweaking the kernel and network configurations, and disabling OpenWrt's native firewall system.

Kernel Parameters and a Network Tweak

Recompiling the OpenWrt image with CONFIG_BRIDGE_NETFILTER=y set in the Linux kernel is the first of two steps in enabling iptables' bridging mode in OpenWrt. The second step is either to delete the following parameters in /etc/sysctl.conf or set each of them to "1" rather than "0":

```
net.bridge.bridge-nf-call-arptables=0
net.bridge.bridge-nf-call-ip6tables=0
net.bridge.bridge-nf-call-iptables=0
```

In addition, I need to correct an error I made in the OpenWrt network configuration I showed you last time. You may recall that I changed OpenWrt's default configuration, such that all Ethernet ports were assigned to a single VLAN and bridge.

But possibly due to the way the Linux kernel interacts with the bridge hardware on my Linksys WRT54GL, with that configuration, I find that iptables ignores inter-VLAN traffic—that is, traffic between ports on the same VLAN. In order to get iptables to work properly on this hardware and on OpenWrt, I actually need two VLANs: one corresponding to my

OpenWrt Performance as a Transparent Firewall

In researching this article, I had a nasty surprise. Although in the past I had seen articles and how-tos on making transparent firewalls with OpenWrt, this mode of operation is not supported by default in the Kamikaze and Backfire releases. Reportedly, running iptables in bridging mode under OpenWrt reduces overall system performance by a whopping 40%!

I proceeded writing this series anyhow, because I wanted to see for myself just how big an effect this is, and it seemed to me that the series still would be useful just for the sake of explaining how to install and use OpenWrt, and for explaining how to write iptables rules for transparent firewalls. However, at several points, I've written of my doubts as to the example OpenWrt/WRT54GL's suitability for high-bandwidth/high-availability settings.

Also, hopefully without sounding *too* grandiose, I hoped that by spurring greater interest in OpenWrt's flawed capability, I might encourage someone to get to the bottom of *why* OpenWrt performance plunges when run with iptables in bridging mode. Surely there's a reason that this not terribly new kernel feature is problematic in OpenWrt!

I say all this because I want to be clear that although transparent Linux firewalls in general constitute an interesting and useful technology, the specific combination of a $65 broadband router plus OpenWrt running in this mode is probably suitable *only* in a home or lab setting, not for any situation where you need to move large volumes of packets very quickly and very reliably (which is hopefully unnecessary for me to say, given that the WRT54GL is marketed to home users in the first place). I say it also so you understand why you have to go through the hoops of recompiling the OpenWrt image and editing /etc/sysctl.conf to get iptables bridging working in OpenWrt.

"uplink" (the Ethernet port connected to the outside world) and my "LAN" (everything else). These two VLANs, however, are still associated with the same bridge interface.

To create a separate VLAN for my uplink port, which is my WRT54GL's "WAN" port (or "port 4" to OpenWrt), I issue these commands on my router:

```
root@sugartongs:/etc/config# uci set network.eth0_1=switch_vlan
root@sugartongs:/etc/config# uci set network.eth0_1.device="eth0"
root@sugartongs:/etc/config# uci set network.eth0_1.vlan="1"
root@sugartongs:/etc/config# uci set network.eth0_1.ports="4 5"
```

(Port 5, you'll recall, is a virtual port associated with the kernel, that must be included in *all* "ports" statements in OpenWrt network configurations, which is why our "...ports" statement is set to "4 5" rather than just "4".)

To remove the WAN port from the other VLAN (eth0_0) I set up last time, I use this command:

```
root@sugartongs:/etc/config# uci set network.eth0_0.ports="0 1 2 3 5"
```

Next, in my bridge configuration, for the network I named "lan", I associate both VLANs with the bridge:

```
root@sugartongs:/etc/config# uci set
➥network.lan.ifname="eth0.0 eth0.1"
```

And finally, I list my new network configuration to make sure everything's correct, commit the changes and reboot:

```
root@sugartongs:/etc/config# uci show network
root@sugartongs:/etc/config# uci commit
root@sugartongs:/etc/config# reboot
```

Listing 1 shows what the resulting /etc/config/network file looks like.

Note that on your system, sections may be listed "out of order", for example, with one VLAN section near the top and another near the bottom. Commands *within* a given section need to be in the correct order, but the sections *themselves* do not, so don't worry!

Listing 1. Corrected /etc/config/network

```
config 'switch' 'eth0'
        option 'enable' '1'

config 'switch_vlan' 'eth0_1'
        option 'device' 'eth0'
        option 'vlan' '1'
        option 'ports' '4 5'

config 'switch_vlan' 'eth0_0'
        option 'device' 'eth0'
        option 'vlan' '0'
        option 'ports' '0 1 2 3 5'

config 'interface' 'loopback'
        option 'ifname' 'lo'
        option 'proto' 'static'
        option 'ipaddr' '127.0.0.1'
        option 'netmask' '255.0.0.0'

config 'interface' 'lan'
        option 'type' 'bridge'
        option 'proto' 'static'
        option 'netmask' '255.255.255.0'
        option 'ipaddr' '10.0.0.253'
        option 'ifname' 'eth0.0 eth0.1'
```

Disabling OpenWrt's DHCP and Firewall System

You also have to disable OpenWrt's native DHCP and iptables systems. The need for disabling DHCP services is obvious: acting as a DHCP server wouldn't be very "transparent" behavior! So, disable it with these two commands:

```
root@sugartongs# /etc/init.d/dnsmasq disable
root@sugartongs# /etc/init.d/dnsmasq stop
```

OpenWrt's native iptables script (/etc/init.d/firewall) is fine if you want to use OpenWrt as a standard "Layer 3" (routing) firewall. Leaving this script enabled allows you to use the uci command and the file /etc/config/firewall to manage iptables in a manner very similar to how you manage network configuration and other OpenWrt system settings.

However, this system doesn't lend itself very well to running iptables in bridging mode—to use it that way, you'd need to hack the script extensively, which would be a bewildering task given the large number of custom tables it uses beyond "INPUT", "OUTPUT" and "FORWARDING". Therefore, disable it like this:

```
root@sugartongs# /etc/init.d/firewall disable
root@sugartongs# /etc/init.d/firewall stop
```

Now you can create a custom iptables script more suitable for a transparent firewall.

Example Network Topology

In order to write a firewall script, you need to consider your network's topology and how the transparent firewall fits in. Figure 1 shows the example home network I sketched out in Part II of this series, with a firewall cabled between the network's Internet uplink (via DSL router or cable modem) and its backbone (which collapses back to a wireless broadband

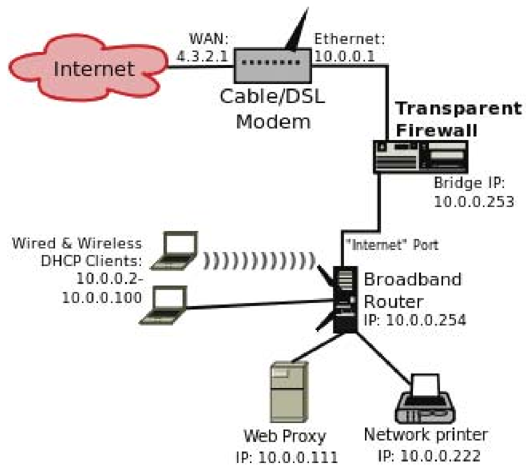


Figure 1. Example Home Network

router configured with Internet uplink and LAN on the same logical subnet).

You could use a number of topologies instead. If you have only a few hosts on your internal network, and your Internet uplink device is already providing DHCP services, you could use your transparent firewall as your broadband router (though configuring WLAN on OpenWrt is outside this series' scope). If your cable modem or DSL router includes a switch and/or wireless LAN access point, you could connect some of your network nodes directly to that and use your transparent firewall to protect other devices.

I'm going to stick with the topology in Figure 1, however, for simplicity's sake. It should be clear enough how to customize my sample iptables script for whatever topology you choose. Let's take a closer look at Figure 1.

In order to write a firewall script, you need to consider your network's topology and how the transparent firewall fits in.

The first thing you should notice is that everything on this network resides on the same logical subnet (10.0.0.0/24) except, of course, for the cable/DSL modem's WAN interface (the one connected to the Internet), which has the Internet-routable address 4.3.2.1. That WAN address is strictly illustrative; in actual practice, WAN IP addresses in *any* residential Internet scenario are assigned by your Internet service provider, often automatically, so please don't attempt to set yours to 4.3.2.1!

Another important point is that on this example network, client PCs are assigned IP addresses via DHCP from the pool 10.0.0.2 through 10.0.0.100.

My diagram doesn't indicate which host is providing DHCP services. Is it the cable/DSL modem, the broadband router or the Web proxy?

As a matter of fact, it doesn't matter! Because this entire network fabric is switched, DHCP requests will propagate freely, including through the transparent firewall. However, if the cable/DSL modem acts as the DHCP server, you will need to write rules on the firewall to allow DHCP through in both directions.

Logical Firewall Design

Now that you understand what the network looks like, let's decide how to manage its dataflows. In my example scenario, the firewall will have a "deny by default" policy, as any good firewall should. The task, therefore, will be one of anticipating and allowing the dataflows you need the firewall to allow.

First, assuming the LAN's DHCP server is upstream of the firewall, you need to allow DHCP traffic between UDP port 67 (the DHCP server port) and UDP port 68 (the DHCP client port).

Next, you don't want to lock yourself out of the firewall itself! You need to allow traffic from the LAN to TCP port 22 on the firewall.

As you can see in Figure 1, the example network has an outbound Web proxy. Because one of the best uses of a firewall is to enforce use of a Web proxy, you'll for sure want to allow only outbound Web traffic originating from the Web proxy. You'll also allow outbound DNS queries (and corresponding replies).

That's it! Things downstream of the firewall—that is, transactions between hosts connected to the broadband router shown in Figure 1—don't need to be allowed by the firewall. For example, print jobs sent from wired and wireless DHCP clients to the network printer don't need an "allow LPR" rule, because those packets should never reach the transparent firewall in the first place.

(If, however, you have only a few hosts on your LAN and elect to omit the downstream switch or broadband router and cable them directly to the transparent firewall, this will not be the case. You *will* need to allow for "LAN-to-LAN" transactions of that type.)

Creating a Custom Firewall Script

Now, finally, you're ready to write a custom firewall script! You could, of course, simply edit the file /etc/init.d/firewall. But, that would make it harder to revert to OpenWrt's native uci-driven firewall system later—better to leave that script alone. I prefer to create a new script from scratch, arbitrarily named /etc/init.d/iptables.custom.

Listing 2 shows what /etc/init.d/iptables.custom needs to look like in order to implement the firewall policy we arrived at in the previous section. Let's

Listing 2. Custom iptables Startup Script

```
#!/bin/sh /etc/rc.common
# Customized iptables script for OpenWrt 10.03

START=46

IPTABLES=/usr/sbin/iptables
LOCALIP=10.0.0.253
LOCALLAN=10.0.0.0/24
WEBPROXY=10.0.0.111

stop() {
        echo "DANGER: Unloading firewall's Packet Filters!"
        $IPTABLES --flush
        $IPTABLES -P INPUT ACCEPT
        $IPTABLES -P FORWARD ACCEPT
        $IPTABLES -P OUTPUT ACCEPT
}

start() {
          echo "Loading custom bridging firewall script"

          # Flush active rules, custom tables
          $IPTABLES --flush
          $IPTABLES --delete-chain

          # Set default-deny policies for all three default tables
          $IPTABLES -P INPUT DROP
          $IPTABLES -P FORWARD DROP
          $IPTABLES -P OUTPUT DROP

          # Don't restrict loopback (local process intercommunication)
          $IPTABLES -A INPUT -i lo -j ACCEPT
          $IPTABLES -A OUTPUT -o lo -j ACCEPT

          # Block attempts at spoofed loopback traffic
          $IPTABLES -A INPUT -s $LOCALIP -j DROP

          # pass DHCP queries and responses
          $IPTABLES -A FORWARD -p udp --sport 68 --dport 67 -j ACCEPT
          $IPTABLES -A FORWARD -p udp --sport 67 --dport 68 -j ACCEPT

          # Allow SSH to firewall from the local LAN
          $IPTABLES -A INPUT -p tcp -s $LOCALLAN --dport 22 -j ACCEPT
          $IPTABLES -A OUTPUT -p tcp --sport 22 -j ACCEPT

          # pass HTTP and HTTPS traffic only to/from the web proxy
          $IPTABLES -A FORWARD -p tcp -s $WEBPROXY --dport 80 -j ACCEPT
          $IPTABLES -A FORWARD -p tcp --sport 80 -d $WEBPROXY -j ACCEPT
          $IPTABLES -A FORWARD -p tcp -s $WEBPROXY --dport 443 -j ACCEPT
          $IPTABLES -A FORWARD -p tcp --sport 443 -d $WEBPROXY -j ACCEPT

          # pass DNS queries and their replies
          $IPTABLES -A FORWARD -p udp -s $LOCALLAN --dport 53 -j ACCEPT
          $IPTABLES -A FORWARD -p tcp -s $LOCALLAN --dport 53 -j ACCEPT
          $IPTABLES -A FORWARD -p udp --sport 53 -d $LOCALLAN -j ACCEPT
          $IPTABLES -A FORWARD -p tcp --sport 53 -d $LOCALLAN -j ACCEPT

          # cleanup-rules
          $IPTABLES -A INPUT -j DROP
          $IPTABLES -A OUTPUT -j DROP
          $IPTABLES -A FORWARD -j DROP
}
```

dissect it.

First, note the includes file /etc/rc.common at the top: this provides functions like enable, disable and other housekeeping functions that OpenWrt uses to manage startup files.

Next, `START=46` specifies the priority/order for running this script at startup. `46` is the same slot that the default OpenWrt "firewall" startup script uses, which is to say, after networking is enabled but before the DropBear SSH server and other network services are started.

Next come some "shorthand" variables we'll use throughout the script. `IPTABLES`, obviously enough, specifies the full path to the local iptables command. `LOCALIP` is the firewall's bridge IP address; `LOCALLAN` is the network address of the local LAN, and `WEBPROXY` gives the IP address of the Web proxy.

The "stop" function (as in `./iptables.custom stop`) causes the script to flush all iptables rules from kernel memory and to load default `ACCEPT` policies for all three default firewall tables, `INPUT`, `FORWARD` and `OUTPUT`. This does *not* "stop all traffic"; rather, it stops all *restrictions* on traffic (thus, the warning message).

Now we come to the heart of the script: the "start" function, containing the firewall policy in the form of a list of iptables commands.

First, flush any active rules and delete any custom tables, so you begin with a clean slate (`$IPTABLES --flush` and `$IPTABLES --delete-chain`). Next, set default deny policies for the INPUT, FORWARD and ACCEPT chains. (You could just as easily choose `REJECT` as the default policy, but because this involves sending ICMP replies to jilted clients, versus DROP's simply ignoring them, there's a slight performance benefit to DROP.)

Next come two rules to allow interprocess communication on the firewall itself, by allowing all packets arriving from and destined for the "loopback" interface. This is followed immediately, however, by an antispoofing rule that blocks traffic addressed to the firewall *from* the firewall's own IP address.

Next are two rules allowing DHCP requests—that is, packets from UDP port 68 sent to UDP port 67—and DHCP responses—that is, packets from UDP port 67 to UDP port 68. These two rules are necessary only if your DHCP server is on the other side of your firewall from your DHCP clients.

You may have noticed that these two DHCP rules and the subsequent rules for SSH, HTTP proxying and DNS are "stateless". Rather than invoking the iptables "state" module, which lets you allow, for example, outbound DHCP queries while letting the kernel decide what constitutes a valid response, you're explicitly allowing the reply traffic. This is an admittedly archaic way to write iptables rules.

However, as I mentioned in the sidebar, OpenWrt has significant performance issues when used as a bridging firewall. Because the "state" module imposes still more of a performance hit, and because this firewall policy is simple to begin with, I'm doing it the old-fashioned way. For a bridging firewall on a better-performing distribution/hardware combination, I definitely would take advantage of Linux's state-tracking features!

For a bridging firewall on a better-performing distribution/hardware combination, I definitely would take advantage of Linux's state-tracking features!

The next pair of rules in Listing 2 allows SSH connections to the firewall itself, but only from the local LAN. Note that the "incoming" leg of SSH transactions is handled in the INPUT table, whereas the "outbound" leg is processed in the OUTPUT table. If you were using `-m state`, the OUTPUT leg would be implicit.

Next come two pairs of rules allowing only the Web proxy to send and receive traffic to/from TCP ports 80 and 443, which, of course, correspond to HTTP and HTTPS, respectively.

This wouldn't work unless DNS did also, so next are rules allowing DNS queries to TCP and UDP ports 53 (ordinarily, DNS queries just use UDP, but once in a while they can occur over TCP as well).

Finally, the script ends with three "cleanup" rules that place a "drop all" rule at the bottom of each of the default tables. These are, of course, redundant with the default "DROP" policies I set near the beginning of the `start()` function, but specifying such cleanup rules are a firewall best practice; sometimes redundancy is desirable!

When you type in any firewall script, be careful! At the very least, double- and triple-check the SSH rules that allow access to the firewall. If there's any problem with those rules, you'll be locked out once you run the script, and you may even need to re-flash your firewall to recover. You can fix other things if SSH works, but if SSH doesn't work, you'll be stuck.

Once you're confident enough to test your rules, save the new script. Be sure to set the "execute" bit on it like so:

```
root@sugartongs:/etc/init.d# chmod a+x ./iptables.custom
```

And, enable the script at startup, like this:

```
root@sugartongs:/etc/init.d# ./iptables.custom enable
```

Now for the moment of truth—load the rules:

```
root@sugartongs:/etc/init.d# ./iptables.custom start
```

Test the rules by making sure the things you want to work still do (connecting back to the firewall via SSH, surfing the Web via your Web proxy and so forth). Also, be sure to test some things you don't expect to work, such as surfing the Web *without* going through the proxy or connecting to an FTP server using an FTP client application. In my own experience, the challenge with OpenWrt is getting iptables to "see" and act on traffic; the real test is ensuring that it's blocking *anything*!

Conclusion

And with that, I've completely filled up this month's allotted space. I'll wrap up the series next month with some tips and tricks, and a suitably flowery "Conclusion" paragraph that I promise will be much more worthwhile than this one. For now, I'll simply say, "good luck!"■

Mick Bauer (darth.elmo@wiremonkeys.org) is Network Security Architect for one of the US's largest banks. He is the author of the O'Reilly book *Linux Server Security*, 2nd edition (formerly called *Building Secure Servers With Linux*), an occasional presenter at information security conferences and composer of the "Network Engineering Polka".

Resources

Home Page for the OpenWrt Project: **www.openwrt.org**

OpenWrt's Unified Configuration Interface Documentation: **wiki.openwrt.org/doc/uci**

The OpenWrt Forum (where you'll end up asking for help sooner or later, if you use OpenWrt more than very casually): **https://forum.openwrt.org**

HARDWARE

ZOTAC ZBOX HD-ID11

With the arrival of the NVIDIA ION GPU, you can build a media server that will fit in the palm of your hand. And, of course, it runs Linux. STEVEN EVATT

For years I have toyed with the idea of setting up a media server for my entertainment center. The challenges in my way included cost, features, aesthetics, background noise and user-friendliness. All of those things are important because I'm not the only person who will be using the system I build. Hardware and software technologies are coming together to address all these challenges. With the advent of the Intel Atom processor and the NVIDIA ION GPU, affordable hardware now is available that allows for an HDTV media server. Likewise, software, such as XBMC and Boxee, has matured and provides a fun and friendly user experience for all levels of users.

For $249.99, the HD-ID11 is small, quiet and looks slick. The chassis is all plastic and feels a bit flimsy when deconstructed. However, the plastic is fairly thick and has tight tolerances. When the cover is in place and set screws tightened, it feels solid.

The front of the HD-ID11 has a 3.5mm headphone jack, a 3.5mm microphone jack, a USB port, an SD card reader, a power button and activity LEDs. The top of the case has a large blue O that lights up while the machine is on. It looks nice, but it can be turned off in the BIOS if it is bothersome. On the side, there is a USB port with a rubber stopper, and on the back, there are four more USB ports. The back also sports HDMI and DVI outputs, 10/100/1000 Ethernet, eSATA, optical out and a port for the power brick.

Hardware

What makes the ZOTAC ZBOX HD-ID11 special is all the power that's packed into the small package. The machine is only 7.4" x 7.4" x 1.73" (188mm x 188mm x 44mm). Here are some of the main technical specs:

- CPU: Intel ATOM D510 (dual-core, 1.66GHz), 667MHz front-side bus.
- Chipset: Intel NM10 Express chipset.
- GPU: NVIDIA ION GPU (with 512MB DDR3 memory).
- Networking: Gigabit (10/1000/10000 Mbps), 802.11b/g/n.
- Audio: onboard 8-channel digital audio, stereo analog audio.
- I/O: HDMI, DVI (DVI-to-VGA dongle included), S/PDIF, mic/headphone, 6 x USB 2.0, RJ45, eSata.
- Memory slot: 1 x 200-pin DDR2-800 SO-DIMM slot.
- Hard drive slot: 1 x 2.5" hard drive (SATA 3.0Gb/s).

One thing separating the ZBOX from the competition is that it does not ship with memory or a hard drive. This allows you to tailor the computer to your needs without buying too much hardware or paying an inflated price for those components.

The HD-ID11 has support for up to 4GB of memory by using a single 200-pin DDR2-800 memory module. I installed 2GB of Kingston DDR2 RAM, which performed flawlessly. If you plan on using the ZBOX for playback only, 1GB of RAM should be sufficient. With the extra RAM available, I decided to use a 640GB hard drive and went with local management for the media. The Intel Atom processor is powerful enough to do a good job with video playback (via the NVIDIA ION GPU) and manage the library at the same time.

BIOS

The ZBOX uses a standard American Megatrends BIOS that can be entered by pressing the Delete key during the boot phase. The settings I felt worth changing included the boot priority, turning off the ZBOX logo at boot time and having the ZBOX restart after a power failure. The other BIOS settings had sane defaults.

The first thing I noticed when I booted the ZBOX with Ubuntu was it did not take long before the CPU fan would spin up to maximum and start to sound like a jet engine. This was worrisome, as I

Figure 1. The ZOTAC ZBOX HD-ID11 with dual-core Atom D510 CPU, NVIDIA ION GPU and HDMI output makes a great low-cost home-theater media box.

intended to keep the ZBOX in my living room. Fortunately, there is a BIOS update available to fix this problem. As with most BIOS updaters, the updater used by the ZBOX requires a DOS boot disk to run. See **wiki.fdos.org/Installation/BootDiskCreateUSB** for some easy-to-follow instructions for creating a free DOS USB boot drive. Once the free DOS image is booted, you can switch to the drive with your BIOS flasher and follow the updater instructions.

OS Installation

Installing Debian on the box was the only time the Intel Atom processor felt slow. This step took more than twice as long as I was expecting. After booting into Ubuntu via the USB memory stick, I formatted my internal hard drive and ran debbootstrap to install Debian Squeeze on the hard drive. Once debbootstrap is complete, do not forget to fix the fstab, networking and install GRUB before rebooting.

H.264 Decode Acceleration

The main reason to opt for a Atom/ION box is for watching high-definition content. The ION GPU supports full hardware decode acceleration of all H.264 content

Figure 2. Back panel of the HD-ID11 has eSATA, four USB, 10/100/1000 Ethernet, DVI and HDMI outputs, optical out and a connector for power.

Figure 3. The front of the HD-ID11 has a 3.5mm headphone jack, 3.5mm microphone jack, SD card reader, USB port, activity LEDs and power button.

Figure 4. HD-ID11 motherboard with 2GB of Kingston RAM installed. Mount and thumbscrew is for the 2.5" HDD/SSD.

(1080i/p) with HDMI out. With the right software, you can watch both Blu-ray and ripped BD content.

I installed both XBMC and Boxee on the HD-ID11 to access my content. Both software packages provide a great user experience and give you the ability to play virtually any type of content. They both provide easy-to-use interfaces into your own content and give you access to content available on the Internet. Boxee is a fork of XBMC, adding social networking to your home-theater experience.

My greatest struggle with my HD-ID11 was getting the digital 8-channel audio to work over HDMI. Although instructions are located several places on the Web (including **ubuntuforums.org/showthread.php?p=6589810**), I was not able to get the audio over HDMI to function properly. This was not a deal-breaker, as the box does support audio out via the 3.5mm headphone jack.

Day-to-Day Use

I have used the HD-ID11 for about a month to play back my local content and to stream content from the Web. I must admit, I love the experience. With few exceptions, the ZBOX has been able to handle any type of content I've thrown at it from inside XBMC and Boxee.

The only content the ION GPU struggles with is Flash video. According to Anadtech.com, the problem stems from the NVIDIA driver requiring too much data to be copied back and forth between the system memory and the GPU framebuffer. There is not enough bandwidth over the single PCIx1 lane to handle this load, which leads to the video stuttering. Even a 480p window does not play smoothly once full-screened. The good news is that NVIDIA is working on an updated driver to fix this problem.

Conclusion

When I started looking at the ZOTAC ZBOX HD-ID11, I wanted to build an affordable system to watch my high-definition content—one that looked nice, was quiet and user-friendly. With a little work, the HD-ID11 fits the bill. Its sleek design and quiet fans allow it to fit into my entertainment center without being noticed. The combination of the Intel Atom processor with the NVIDIA ION GPU provide all the power necessary to make for an enjoyable entertainment experience. Although I would prefer the sound going over HDMI and better Flash video playback, those are issues that should be addressed via driver updates in the future. I'm enjoying the media box so much, I am planning on purchasing a second ZBOX for my bedroom to give me more access to my content.■

Steven Evatt is an IT manager in Houston, Texas, and has been using Linux for more than 16 years. He is active in the local technology community and regularly can be found at barcamps in Texas and Louisiana. In his spare time, he enjoys playing with Ruby on Rails on his site: pricechirp.com.